US010154092B2

(12) **United States Patent**
Gertner

(10) **Patent No.: US 10,154,092 B2**
(45) **Date of Patent: *Dec. 11, 2018**

(54) **DATA SHARING USING DISTRIBUTED CACHE IN A NETWORK OF HETEROGENEOUS COMPUTERS**

(71) Applicant: **LS CLOUD STORAGE TECHNOLOGIES, LLC**, Longview, TX (US)

(72) Inventor: **Ilya Gertner**, Long Beach, CA (US)

(73) Assignee: **LS CLOUD STORAGE TECHNOLOGIES, LLC**, Longview, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 237 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **14/997,327**

(22) Filed: **Jan. 15, 2016**

(65) **Prior Publication Data**

US 2016/0134702 A1 May 12, 2016

Related U.S. Application Data

(60) Continuation of application No. 13/527,126, filed on Jun. 19, 2012, now abandoned, which is a continuation of application No. 10/382,016, filed on Mar. 5, 2003, now Pat. No. 8,225,002, which is a (Continued)

(51) **Int. Cl.**

G06F 3/06	(2006.01)
H04L 29/08	(2006.01)
G06F 17/30	(2006.01)
G06F 12/0808	(2016.01)

(Continued)

(52) **U.S. Cl.**

CPC ***H04L 67/1097*** (2013.01); ***G06F 3/065*** (2013.01); ***G06F 3/067*** (2013.01); ***G06F 3/0619*** (2013.01); ***G06F 3/0635*** (2013.01); ***G06F 12/0808*** (2013.01); ***G06F 12/128*** (2013.01); ***G06F 17/30569*** (2013.01); ***H04L 67/2842*** (2013.01); *G06F 12/12* (2013.01); *G06F 2212/621* (2013.01); *G06F 2212/69* (2013.01)

(58) **Field of Classification Search**

None

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,400,769	A	8/1983	Kaneda et al.	
4,648,030	A *	3/1987	Bomba	G06F 13/36 711/141

(Continued)

OTHER PUBLICATIONS

Dahlin et al., "Cooperative Caching: Using Remote Client Memory to Improve File System Performance," First Symposium on Operating Systems Design and Implementation (OSDI 1994), 1994, pp. 1-14.

(Continued)

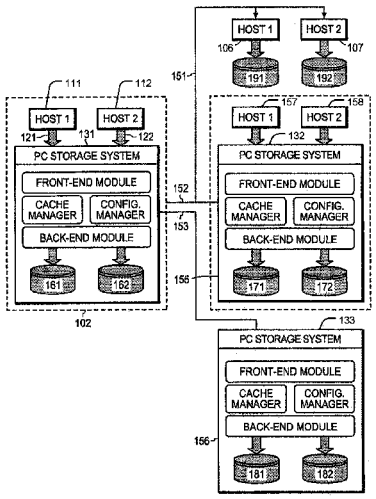
Primary Examiner — Backhean Tiv

(74) *Attorney, Agent, or Firm* — Toler Law Group, PC

(57) **ABSTRACT**

A network of PCs includes an I/O channel adapter and network adapter, and is configured for management of a distributed cache memory stored in the plurality of PCs interconnected by the network. The use of standard PCs reduces the cost of the data storage system. The use of the network of PCs permits building large, high-performance, data storage systems.

24 Claims, 13 Drawing Sheets



Related U.S. Application Data

division of application No. 09/236,409, filed on Jan. 22, 1999, now Pat. No. 6,549,988.

(51) **Int. Cl.**
G06F 12/128 (2016.01)
G06F 12/12 (2016.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,797,813	A *	1/1989	Igarashi	G06F 12/0804	711/118
4,933,846	A *	6/1990	Humphrey	G06F 12/1466	370/463
5,091,412	A	2/1992	Wright et al.		
5,222,224	A *	6/1993	Flynn	G06F 12/0822	711/120
5,251,311	A *	10/1993	Kasai	G06F 12/0817	711/144
5,319,766	A *	6/1994	Thaller	G06F 12/0831	711/145
5,506,975	A	4/1996	Onodera		
5,553,291	A	9/1996	Tanaka et al.		
5,577,204	A *	11/1996	Brewer	G06F 13/1657	710/317
5,577,226	A	11/1996	Percival		
5,598,551	A *	1/1997	Barajas	G06F 12/0835	711/157
5,600,817	A	2/1997	Macon, Jr. et al.		
5,611,049	A	3/1997	Pitts		
5,644,751	A	7/1997	Burnett		
5,649,152	A	7/1997	Ohran et al.		
5,659,794	A *	8/1997	Caldarale	H04L 29/06	709/234
5,701,516	A	12/1997	Cheng et al.		
5,715,455	A	2/1998	Macon, Jr. et al.		
5,717,884	A	2/1998	Gzym et al.		
5,742,792	A	4/1998	Yanai et al.		
5,743,933	A	4/1998	Mellem		
5,748,985	A	5/1998	Kanai		
5,751,993	A	5/1998	Ofek et al.		
5,758,050	A	5/1998	Brady et al.		
5,761,734	A *	6/1998	Pfeffer	G06F 9/52	709/201
5,768,211	A *	6/1998	Jones	G11C 8/16	365/189.04
5,778,353	A	7/1998	Schiefer et al.		
5,787,469	A *	7/1998	Merrell	G06F 12/0897	711/119
5,787,473	A	7/1998	Vishlitzky et al.		
5,790,795	A	8/1998	Hough		
5,802,553	A	9/1998	Robinson et al.		
5,802,569	A *	9/1998	Genduso	G06F 9/3802	711/137
5,805,857	A	9/1998	Colegrove		
5,819,292	A	10/1998	Hitz et al.		
5,819,310	A	10/1998	Vishlitzky et al.		
5,828,475	A	10/1998	Bennett et al.		
5,841,997	A	11/1998	Bleiweiss et al.		
5,848,251	A	12/1998	Lomelino et al.		
5,852,715	A	12/1998	Raz et al.		
5,854,942	A	12/1998	Penokie		
5,860,026	A	1/1999	Kitta et al.		
5,860,137	A	1/1999	Raz et al.		
5,887,146	A	3/1999	Baxter et al.		
5,896,506	A	4/1999	Ali et al.		
5,898,828	A	4/1999	Pignolet et al.		
5,900,015	A *	5/1999	Herger	G06F 12/0824	711/141
5,901,327	A	5/1999	Ofek		
5,913,029	A	6/1999	Shostak		
5,974,503	A *	10/1999	Venkatesh	G06F 11/1076	348/E5.008
6,016,500	A	1/2000	Waldo et al.		
6,021,469	A	2/2000	Tremblay et al.		
6,026,461	A	2/2000	Baxter et al.		
6,044,438	A *	3/2000	Olnowich	G06F 12/0813	707/999.201
6,101,497	A *	8/2000	Ofek	G06F 11/1451	707/657
6,101,508	A *	8/2000	Wolff	G06F 9/52	707/999.001
6,119,151	A *	9/2000	Cantrell	G06F 17/30132	707/E17.01
6,122,659	A	9/2000	Olnowich		
6,151,618	A	11/2000	Wahbe et al.		
6,182,111	B1	1/2001	Inohara et al.		
6,185,609	B1	2/2001	Rangarajan et al.		
6,226,680	B1 *	5/2001	Boucher	G06F 5/10	709/230
6,253,260	B1 *	6/2001	Beardsley	G06F 3/061	710/20
6,256,637	B1	7/2001	Venkatesh et al.		
6,260,077	B1	7/2001	Rangarajan et al.		
6,260,120	B1 *	7/2001	Blumenau	G06F 3/0622	707/999.003
6,311,186	B1 *	10/2001	MeLampy	H04Q 3/0029	
6,341,311	B1 *	1/2002	Smith	G06F 17/30902	707/E17.12
6,389,479	B1 *	5/2002	Boucher	H04L 29/06	709/230
6,397,242	B1	5/2002	Devine et al.		
6,438,652	B1	8/2002	Jordan et al.		
6,457,047	B1	9/2002	Chandra et al.		
6,496,847	B1	12/2002	Bugnion et al.		
6,549,988	B1 *	4/2003	Gertner	G06F 17/30569	707/E17.006
6,711,632	B1 *	3/2004	Chow	G06F 12/0804	709/237
6,785,714	B1	8/2004	Thompson et al.		
6,829,637	B2	12/2004	Kokku et al.		
6,850,980	B1	2/2005	Gourlay		
7,003,587	B1 *	2/2006	Battat	H04L 67/1095	707/E17.005
7,072,056	B1 *	7/2006	Greaves	H04L 51/066	358/1.1
7,133,905	B2	11/2006	Dilley et al.		
7,188,251	B1	3/2007	Slaughter et al.		
7,254,617	B2 *	8/2007	Schuh	G06F 17/30902	707/E17.12
7,287,065	B2	10/2007	Nishi et al.		
7,293,099	B1 *	11/2007	Kalajan	G06F 17/30067	709/203
7,664,883	B2 *	2/2010	Craft	H04L 67/1097	370/463
7,739,379	B1 *	6/2010	Vahalia	G06F 17/30171	709/203
7,864,758	B1 *	1/2011	Lolayekar	H04L 67/1097	370/229
8,225,002	B2 *	7/2012	Gertner	G06F 17/30569	709/217
2001/0013085	A1 *	8/2001	Yamamoto	G06F 3/0626	711/114
2001/0037406	A1 *	11/2001	Philbrick	H04L 29/06	709/250
2002/0002625	A1 *	1/2002	Vange	G06F 9/5027	709/246
2002/0007445	A1 *	1/2002	Blumenau	G06F 3/0622	711/153
2002/0065879	A1	5/2002	Ambrose et al.		
2002/0091844	A1 *	7/2002	Craft	G06F 5/10	709/230
2002/0194294	A1 *	12/2002	Blumenau	G06F 3/0605	709/213
2003/0069889	A1 *	4/2003	Ofek	G06F 11/1451	
2003/0126372	A1 *	7/2003	Rand	G06F 12/0831	711/141
2003/0145114	A1 *	7/2003	Gertner	G06F 17/30569	709/246
2004/0022094	A1 *	2/2004	Radhakrishnan	G06F 12/0813	365/200

(56) **References Cited**

U.S. PATENT DOCUMENTS

2009/0282101	A1	11/2009	Lim et al.	
2012/0259953	A1*	10/2012	Gertner	G06F 17/30569 709/217
2016/0134702	A1*	5/2016	Gertner	G06F 17/30569 709/216
2017/0161189	A1*	6/2017	Gertner	G06F 12/0808

OTHER PUBLICATIONS

NCR 5100M, http://www.hardwood-intl.com/ncrproducts/5100M_NCR.asp, printed Nov. 27, 2006, 5 pages.

Internet Archive Waybackmachine http://archive.org/web/web.php, printed Nov. 27, 2006, 1 page.

Karedla, et al., Caching Strategies to Improve Disk System Performance, Computer, vol. 27, No. 3, Mar. 1994 (Research paper), pp. 38-46, Abstract, 2 pages.

Smith, "Cache Memories," Computing Surveys, vol. 14, No. 3, Sep. 1982, pp. 473-530.

* cited by examiner

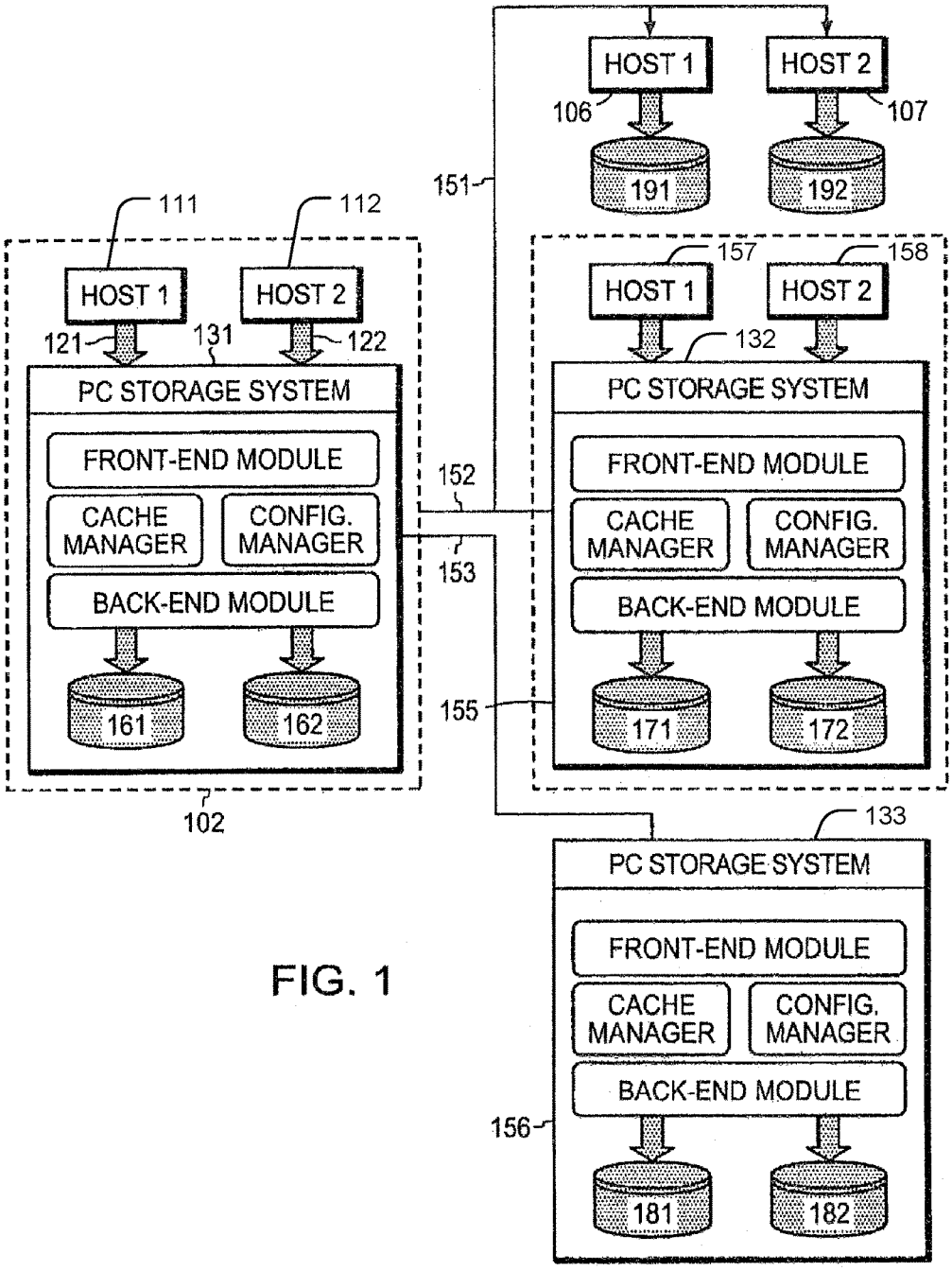

HOST 1
HOST 2
106
107
191
192
151
111
112
HOST 1
HOST 2
121
131
122
PC STORAGE SYSTEM
FRONT-END MODULE
CACHE MANAGER
CONFIG. MANAGER
BACK-END MODULE
161
162
102
157
158
HOST 1
HOST 2
132
PC STORAGE SYSTEM
FRONT-END MODULE
CACHE MANAGER
CONFIG. MANAGER
BACK-END MODULE
171
172
152
153
155
133
PC STORAGE SYSTEM
FRONT-END MODULE
CACHE MANAGER
CONFIG. MANAGER
BACK-END MODULE
156
181
182

FIG. 1

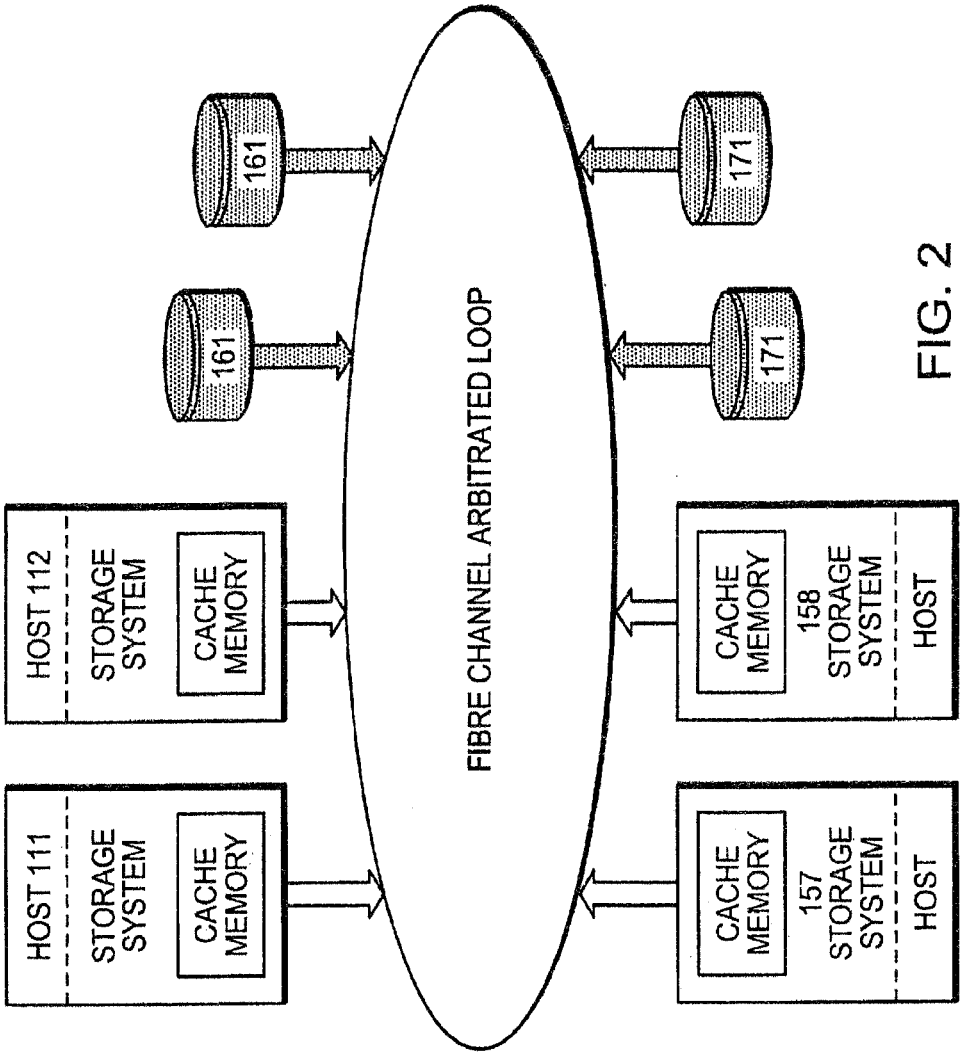

161
161
171
171
FIBRE CHANNEL ARBITRATED LOOP
HOST 112
STORAGE SYSTEM
CACHE MEMORY
HOST 111
STORAGE SYSTEM
CACHE MEMORY
CACHE MEMORY
158 STORAGE SYSTEM
HOST
CACHE MEMORY
157 STORAGE SYSTEM
HOST

FIG. 2

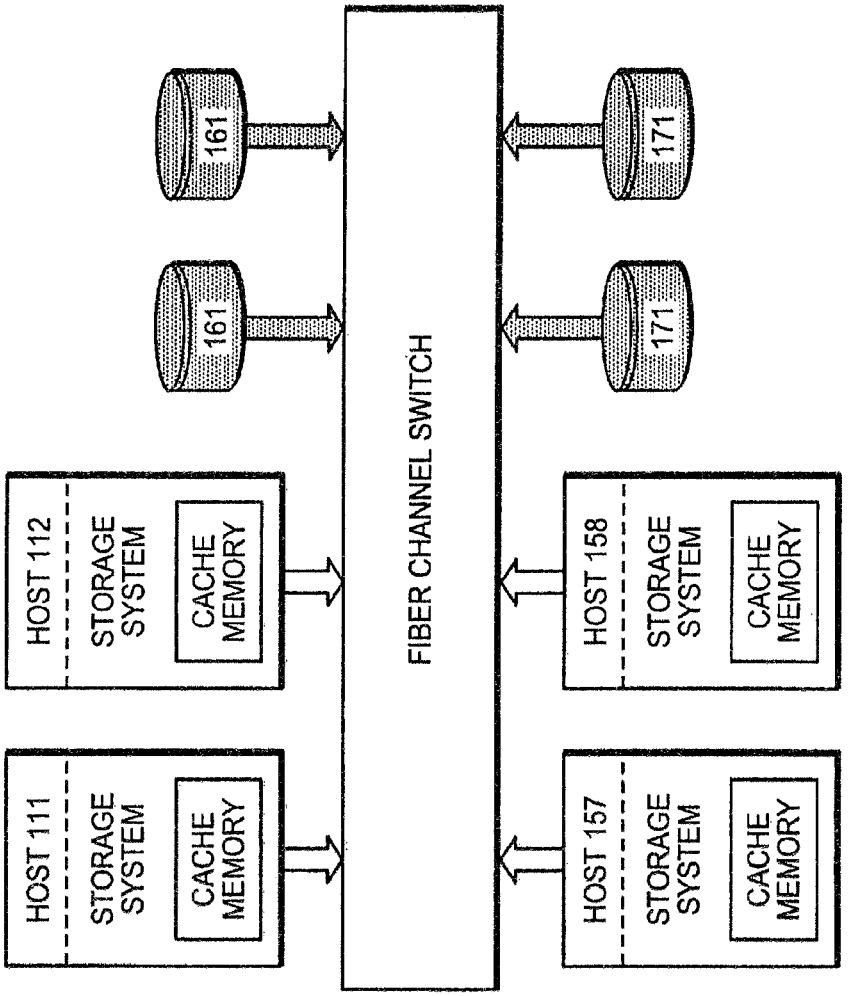

161
161
171
171
FIBER CHANNEL SWITCH
HOST 112
STORAGE SYSTEM
CACHE MEMORY
HOST 111
STORAGE SYSTEM
CACHE MEMORY
HOST 158
STORAGE SYSTEM
CACHE MEMORY
HOST 157
STORAGE SYSTEM
CACHE MEMORY

FIG. 2A

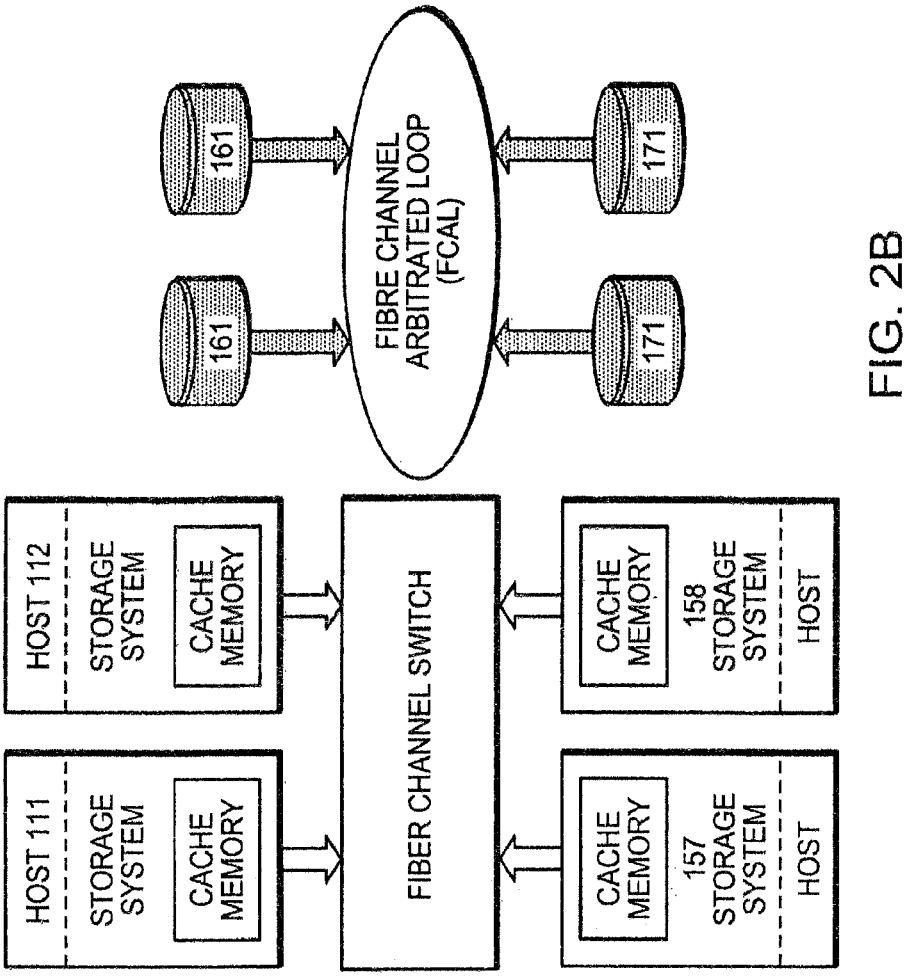

161
161
FIBRE CHANNEL ARBITRATED LOOP (FCAL)
171
171
HOST 112
STORAGE SYSTEM
CACHE MEMORY
HOST 111
STORAGE SYSTEM
CACHE MEMORY
FIBER CHANNEL SWITCH
CACHE MEMORY
158 STORAGE SYSTEM
HOST
CACHE MEMORY
157 STORAGE SYSTEM
HOST

FIG. 2B

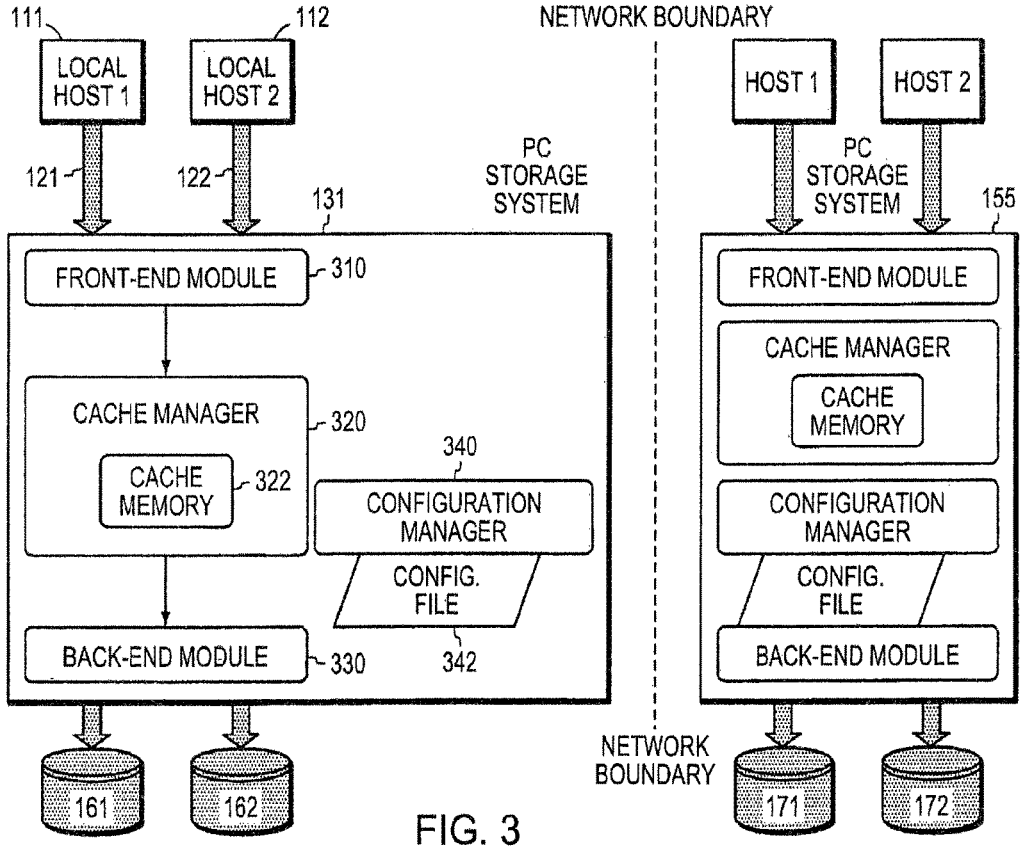

111
112
NETWORK BOUNDARY
LOCAL HOST 1
LOCAL HOST 2
HOST 1
HOST 2
121
122
131
PC STORAGE SYSTEM
PC STORAGE SYSTEM
155
FRONT-END MODULE
310
FRONT-END MODULE
CACHE MANAGER
320
CACHE MANAGER
CACHE MEMORY
322
CACHE MEMORY
340
CONFIGURATION MANAGER
CONFIGURATION MANAGER
CONFIG. FILE
CONFIG. FILE
BACK-END MODULE
330
342
BACK-END MODULE
NETWORK BOUNDARY
161
162
171
172

FIG. 3

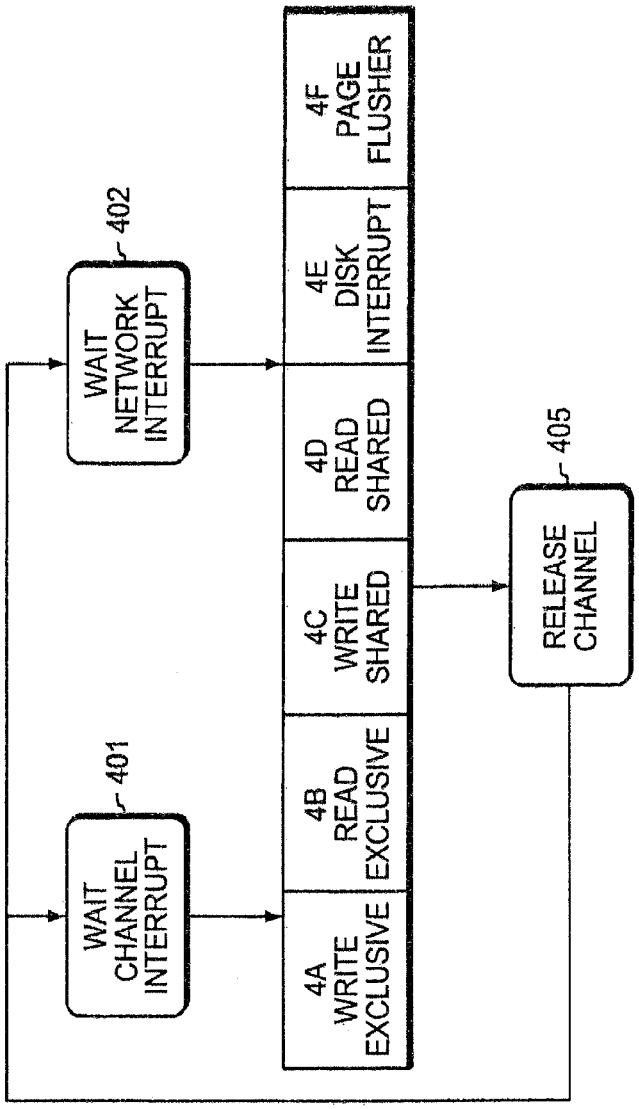
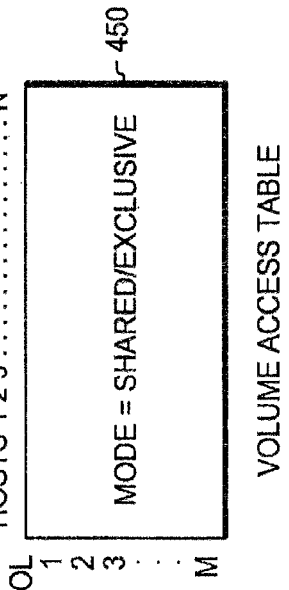

WAIT CHANNEL INTERRUPT
401
WAIT NETWORK INTERRUPT
402
4A WRITE EXCLUSIVE
4B READ EXCLUSIVE
4C WRITE SHARED
4D READ SHARED
4E DISK INTERRUPT
4F PAGE FLUSHER
RELEASE CHANNEL
405

FIG. 4


HOSTS 1 2 3 . . . . . N
VOL
1
2
3
. . .
M
MODE = SHARED/EXCLUSIVE
450
VOLUME ACCESS TABLE

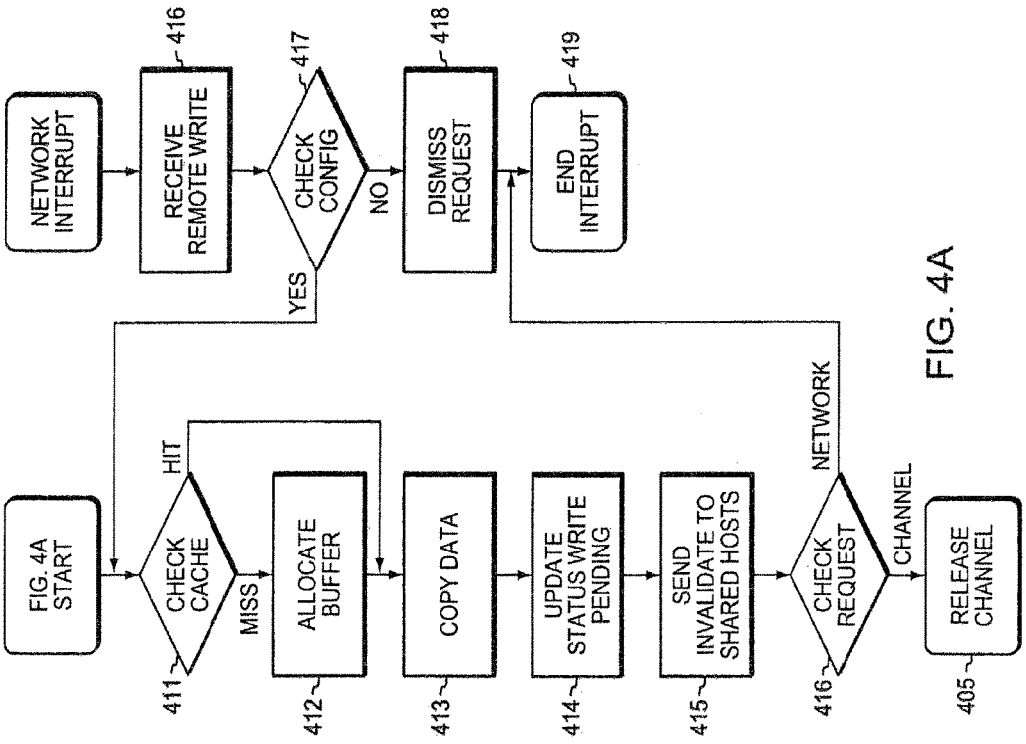


FIG. 4A START
CHECK CACHE
411
HIT
MISS
ALLOCATE BUFFER
412
COPY DATA
413
UPDATE STATUS WRITE PENDING
414
SEND INVALIDATE TO SHARED HOSTS
415
CHECK REQUEST
416
NETWORK
CHANNEL
RELEASE CHANNEL
405
NETWORK INTERRUPT
RECEIVE REMOTE WRITE
416
CHECK CONFIG
417
YES
NO
DISMISS REQUEST
418
END INTERRUPT
419

FIG. 4A

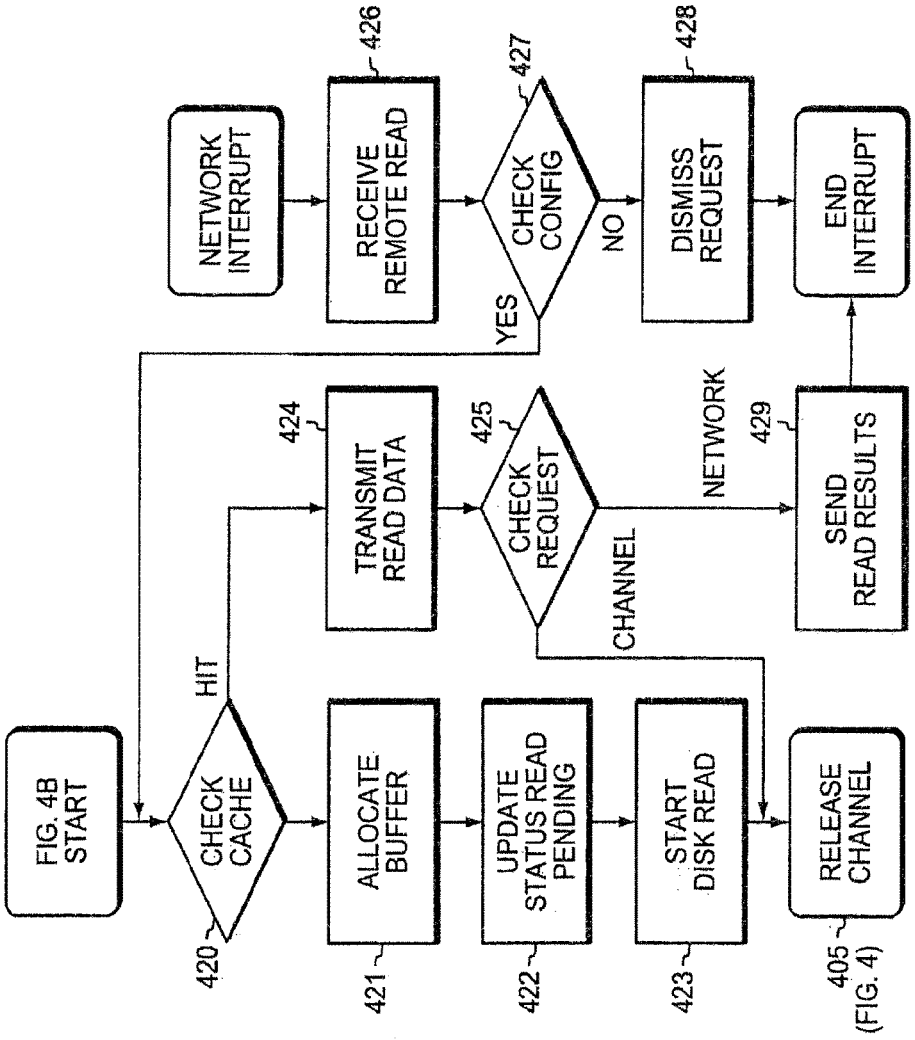


FIG. 4B START
CHECK CACHE
420
HIT
ALLOCATE BUFFER
421
UPDATE STATUS READ PENDING
422
START DISK READ
423
RELEASE CHANNEL
405 (FIG. 4)
TRANSMIT READ DATA
424
CHECK REQUEST
425
CHANNEL
NETWORK
SEND READ RESULTS
429
NETWORK INTERRUPT
RECEIVE REMOTE READ
426
CHECK CONFIG
427
YES
NO
DISMISS REQUEST
428
END INTERRUPT

FIG. 4B

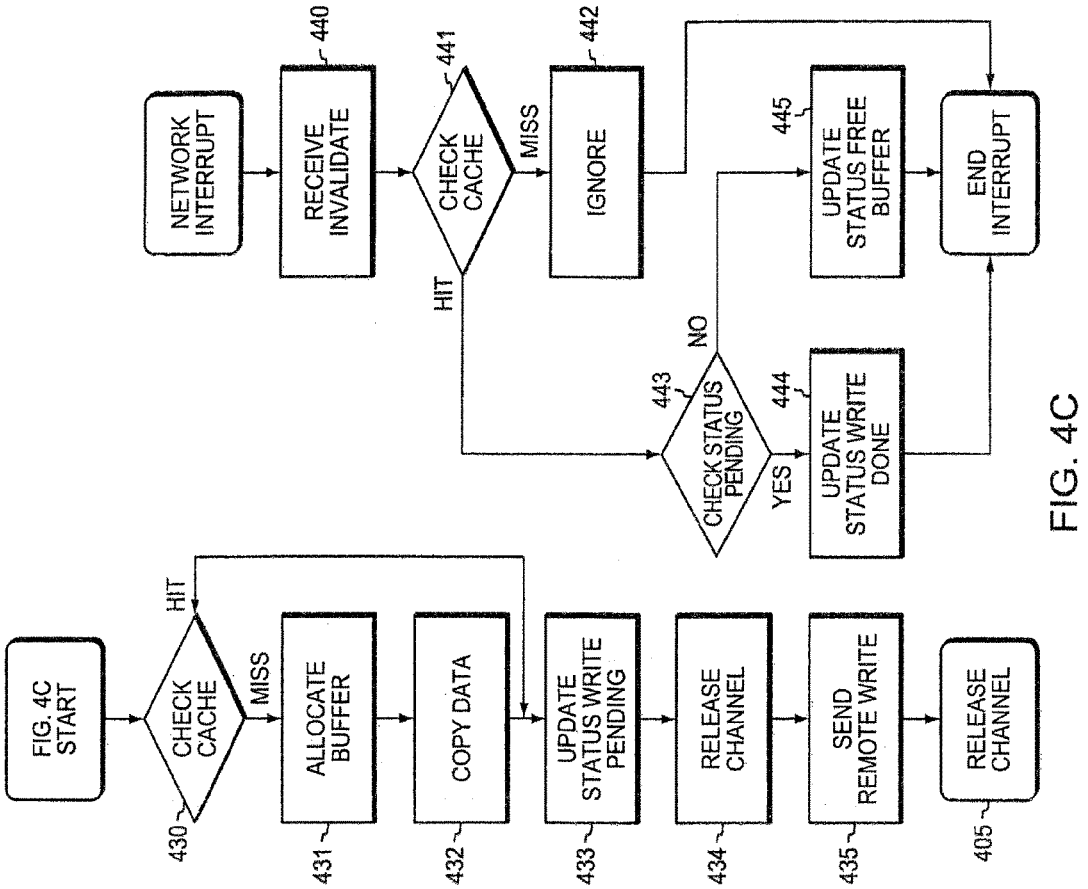


FIG. 4C START
CHECK CACHE
HIT
MISS
430
ALLOCATE BUFFER
431
COPY DATA
432
UPDATE STATUS WRITE PENDING
433
RELEASE CHANNEL
434
SEND REMOTE WRITE
435
RELEASE CHANNEL
405
NETWORK INTERRUPT
RECEIVE INVALIDATE
440
CHECK CACHE
441
HIT
MISS
IGNORE
442
CHECK STATUS PENDING
443
NO
YES
UPDATE STATUS WRITE DONE
444
UPDATE STATUS FREE BUFFER
445
END INTERRUPT

FIG. 4C

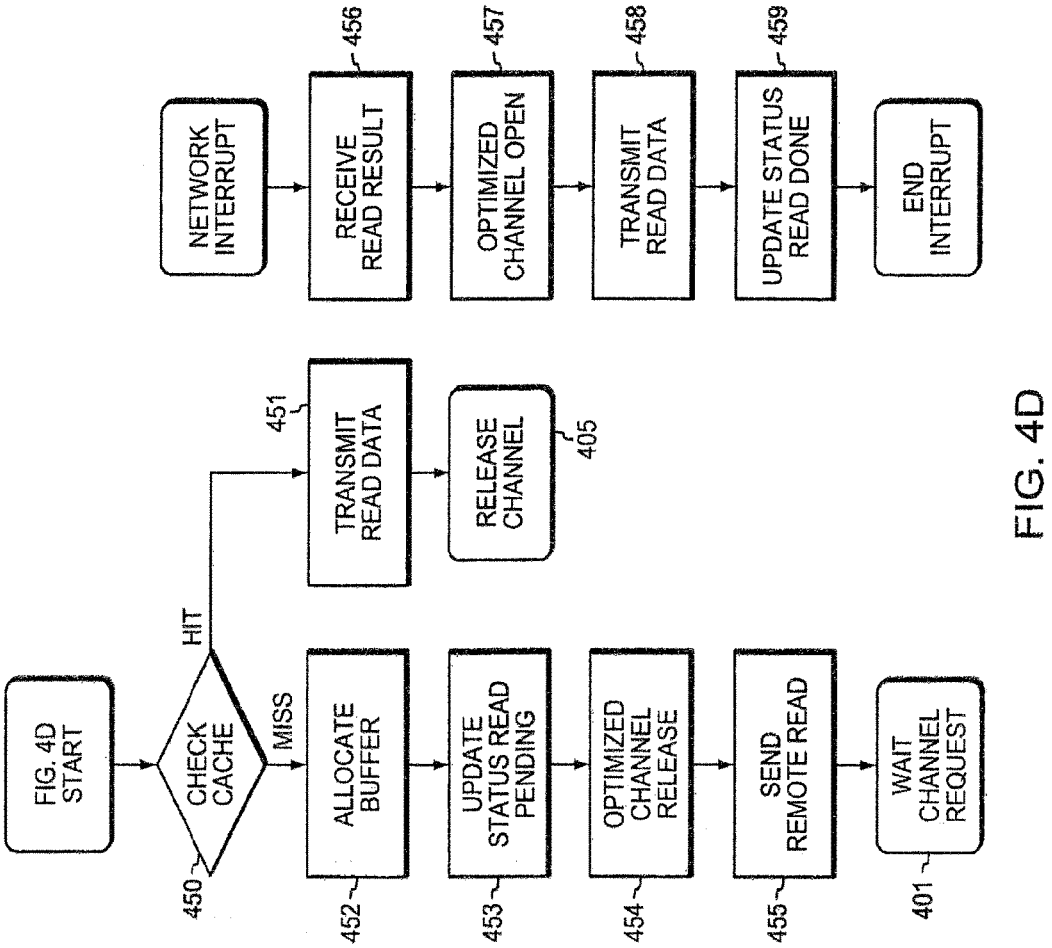


FIG. 4D START
CHECK CACHE
HIT
MISS
450
TRANSMIT READ DATA
451
RELEASE CHANNEL
405
ALLOCATE BUFFER
452
UPDATE STATUS READ PENDING
453
OPTIMIZED CHANNEL RELEASE
454
SEND REMOTE READ
455
WAIT CHANNEL REQUEST
401
NETWORK INTERRUPT
RECEIVE READ RESULT
456
OPTIMIZED CHANNEL OPEN
457
TRANSMIT READ DATA
458
UPDATE STATUS READ DONE
459
END INTERRUPT

FIG. 4D

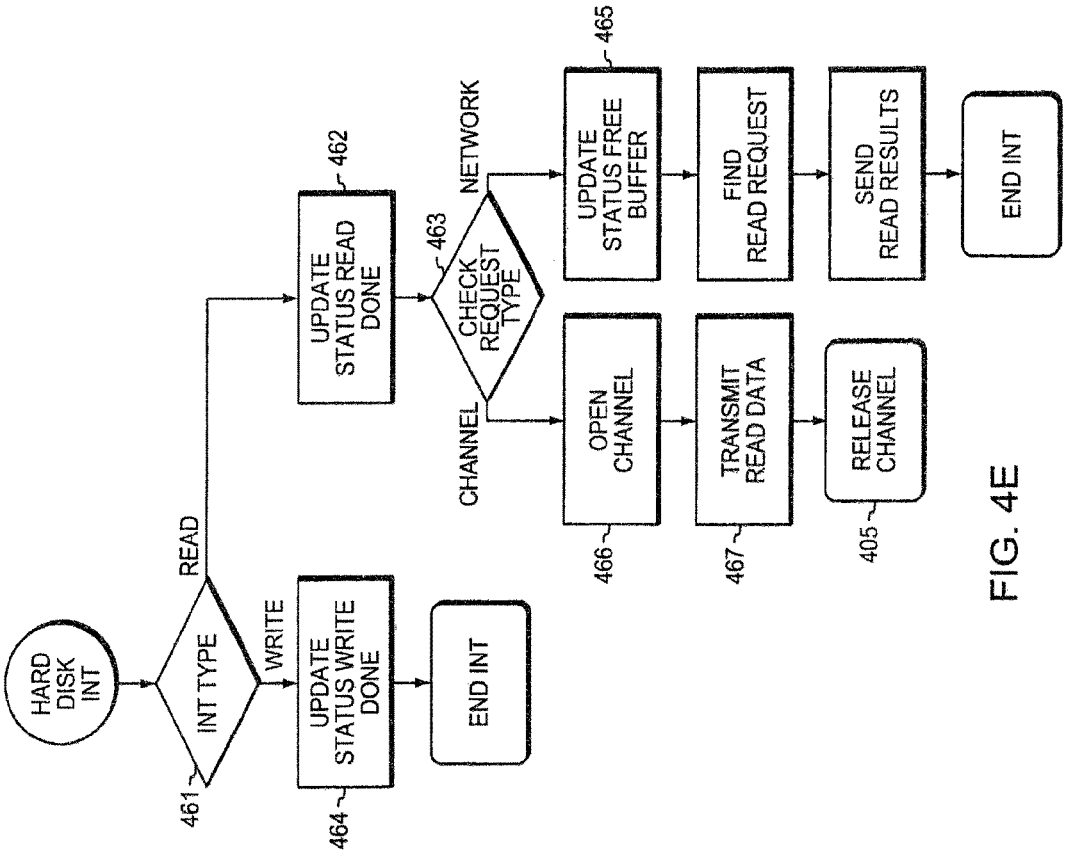

HARD DISK INT
INT TYPE
461
READ
WRITE
UPDATE STATUS WRITE DONE
464
END INT
UPDATE STATUS READ DONE
462
CHECK REQUEST TYPE
463
NETWORK
CHANNEL
UPDATE STATUS FREE BUFFER
465
FIND READ REQUEST
SEND READ RESULTS
END INT
OPEN CHANNEL
466
TRANSMIT READ DATA
467
RELEASE CHANNEL
405

FIG. 4E

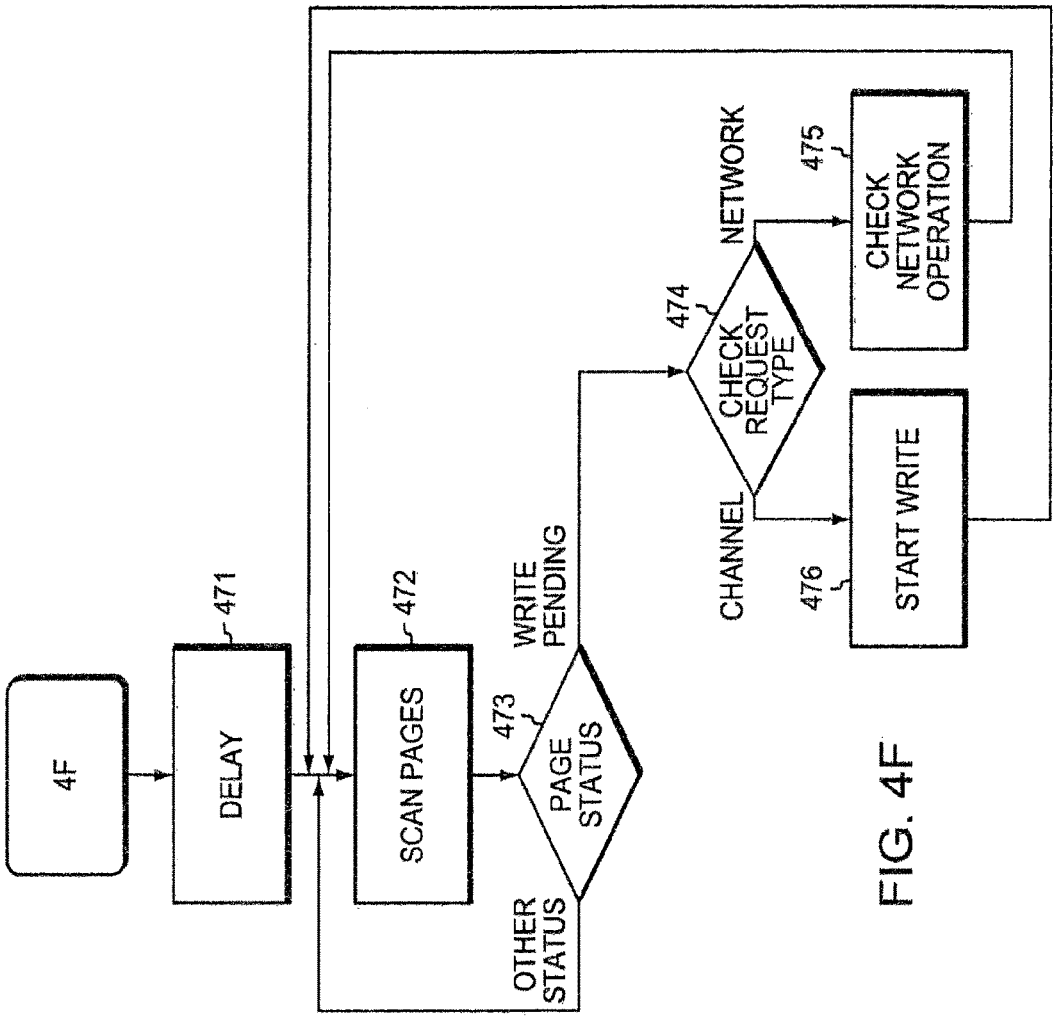

4F
DELAY
471
SCAN PAGES
472
PAGE STATUS
473
WRITE PENDING
OTHER STATUS
CHECK REQUEST TYPE
474
NETWORK
CHANNEL
CHECK NETWORK OPERATION
475
START WRITE
476

FIG. 4F

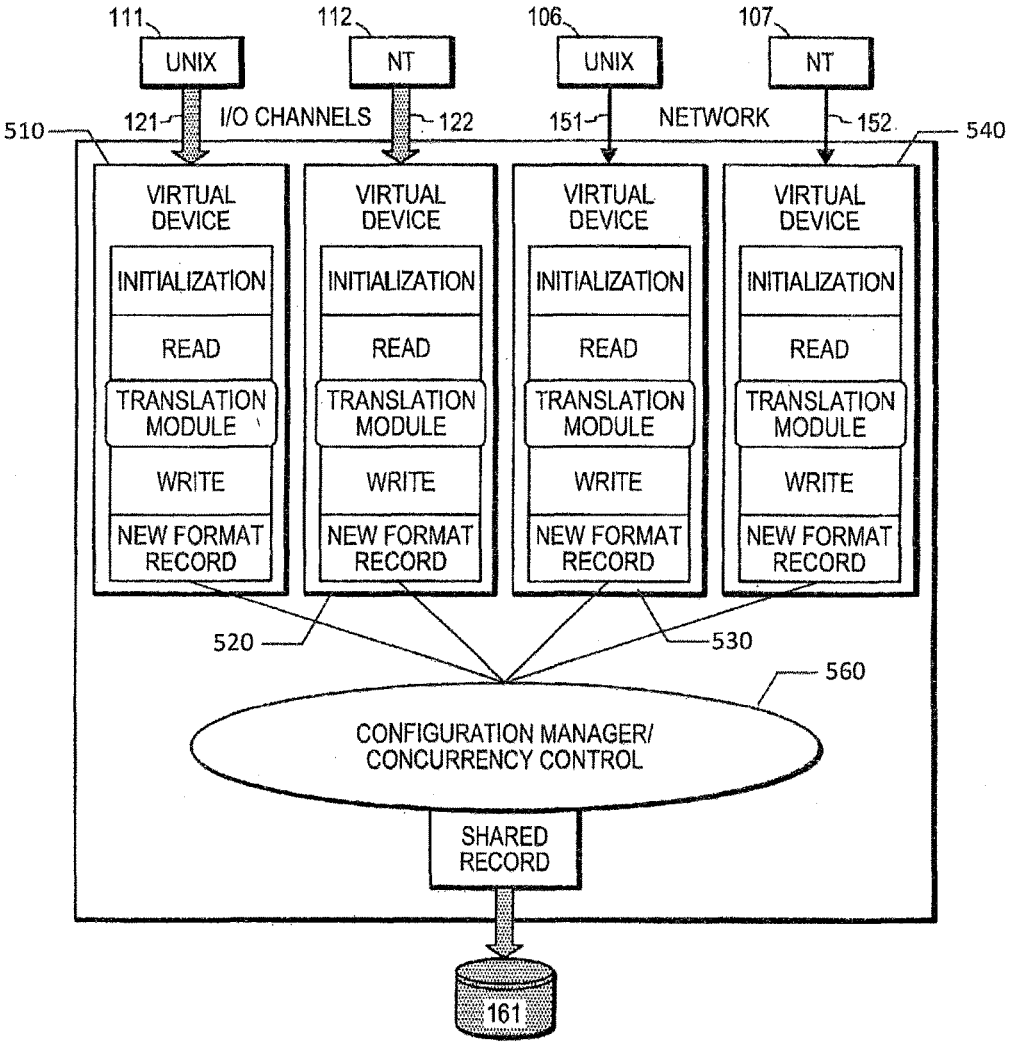

111
UNIX
112
NT
106
UNIX
107
NT
121
I/O CHANNELS
122
151
NETWORK
152
510
540
VIRTUAL DEVICE
INITIALIZATION
READ
TRANSLATION MODULE
WRITE
NEW FORMAT RECORD
VIRTUAL DEVICE
INITIALIZATION
READ
TRANSLATION MODULE
WRITE
NEW FORMAT RECORD
VIRTUAL DEVICE
INITIALIZATION
READ
TRANSLATION MODULE
WRITE
NEW FORMAT RECORD
VIRTUAL DEVICE
INITIALIZATION
READ
TRANSLATION MODULE
WRITE
NEW FORMAT RECORD
520
530
560
CONFIGURATION MANAGER/ CONCURRENCY CONTROL
SHARED RECORD
161

FIG. 5

DATA SHARING USING DISTRIBUTED CACHE IN A NETWORK OF HETEROGENEOUS COMPUTERS

RELATED APPLICATIONS

This application is a continuation of U.S. application Ser. No. 13/527,126, filed Jun. 9, 2012, which is a further continuation of U.S. application Ser. No. 10/382,016, filed on Mar. 5, 2003, now U.S. Pat. No. 8,225,002, issued Jul. 17, 2012, which is a divisional of U.S. application Ser. No. 09/236,409, filed on Jan. 22, 1999, now U.S. Pat. No. 6,549,988, issued Apr. 15, 2003, the entire disclosure of which is incorporated by reference in its entirety.

FIELD OF THE INVENTION

This invention relates generally to the field of cached data storage systems and more particularly to a data storage system that permits independent access from local hosts connected via I/O channels and independent access from remote hosts and remote storage systems connected via network links. A network of PCs permits building a high-performance, scalable, data storage system using off-the-shelf components at reduced cost. A configuration manager ensures consistency of data stored in the distributed cache.

BACKGROUND OF THE INVENTION

A typical data processing system generally involves a cached data storage system that connects to local host computers via I/O channels or remote host computers via network links. The purpose of the data storage system is to improve the performance of applications running on the host computer by offloading I/O processing from the host to the data storage system. The purpose of the cache memory in a data storage system is to further improve the performance of the applications by temporarily storing data buffers in the cache so that the references to those buffers can be resolved efficiently as "cache hits". Reading data from a cache is an order of magnitude faster than reading data from a back end storage device such as a disk. Writing data to a cache is also an order of magnitude faster than writing to a disk. All writes are cache hits because data is simply copied into cache buffers that are later flushed to disks.

Prior art data storage systems are implemented using proprietary hardware and very low-level software, frequently referred to as microcode, resulting in expensive and not portable systems. In contrast to the prior art systems, the preferred embodiment of the present invention uses standard hardware and software components. A network of commercial PCs is used to implement a high-performance data storage system. A method using the network of PCs includes an algorithm for a configuration manager that manages access to the distributed cache memory stored in PCs interconnected by the network. Numerous prior art systems and methods exist for managing cache memory in a data storage system. The prior art has suggested several methods for managing cache for channel attached hosts. U.S. Pat. No. 5,717,884, Gzym, et. al., Feb. 2, 1996, Method and Apparatus for Cache Management, discloses data structures and algorithms that use a plurality of slots, each of which is used to store data files. U.S. Pat. No. 5,757,473, Vishlitzky, et. al., Cache Management system using time stamping for replacement queue, Jul. 28, 1998, discloses a method that uses time stamps to manage queues in a cached data storage system. U.S. Pat. No. 5,751,993, Ofek, et. al., May 12, 1998, Cache Management Systems, discloses yet another aspect in queue management algorithms. U.S. Pat. No. 5,600,817, Macon Jr., et. al., Feb. 4, 1997, Asynchronous read-ahead disk caching using multiple disk I/O processes and dynamically variable prefetch length, discloses read-ahead methods in cached storage systems. U.S. Pat. No. 5,758,050, Brady, et. al., May 26, 1998, Reconfigurable data storage system, discloses a method for reconfiguring a data storage system.

However, the above systems use very specialized embedded operating systems and custom programming in a very low-level programming language such as assembler. The obvious drawback of the above systems is high cost because assembler-level programming is very time consuming. Another drawback is inflexibility and lack of functionality. For example, some features such as reconfigurability in data storage are very limited in proprietary embedded systems when compared to general purpose operating systems. Finally, networking support is very expensive and limited because it relies on dedicated communication links such as T1, T3 and ESCON.

One prior art system using networking of data storage systems is disclosed in U.S. Pat. No. 5,742,792, Yanai, et. al., Apr. 21, 1998, Remote Data Mirroring. This patent discloses a primary data storage system providing storage services to a primary host and a secondary data storage system providing services to a secondary host. The primary storage system sends all writes to the secondary storage system via IBM ESCON, or optionally via T1 or T3 communications link. The secondary data storage system provides a backup copy of the primary storage system. Another prior art system is disclosed in U.S. Pat. No. 5,852,715, Raz, et al., Dec. 22, 1998, System for currently updating database by one host and reading the database by different host for the purpose of implementing decision support functions.

However, the above systems use dedicated communication links that are very expensive when compared to modern networking technology. Furthermore, the data management model is limited to the primary-node sending messages to the secondary node scenario. This model does not support arbitrary read and write requests in a distributed data storage system.

There is a growing demand for distributed data storage systems. In response to this demand some prior art systems have evolved into complex assemblies of two systems, one proprietary a data storage system and the other an open networking server. One such system is described in a white paper on a company web site on Internet. The industry white paper, EMC Data Manager: A high-performance, centralized open system backup/restore solution for LAN-based and Symmetrix resident data, describes two different systems, one for network attached hosts and second for channel attached hosts. The two systems are needed because of the lack of generic networking support. In related products such as Celerra File Server, product data sheets suggest using data movers for copying data between LAN-based open system storage and channel attached storage system.

However, the above systems are built from two systems, one for handling I/O channels, and another for handling open networks. Two systems are very expensive even in minimal configuration that must include two systems.

In another branch of storage industry, network attached storage systems use network links to attach to host computers. Various methods for managing cache memory and distributed applications for network attached hosts have been described in prior art. U.S. Pat. No. 5,819,292, Hitz, et. al., Method for maintaining consistent states of a file system and for creating user-accessible read-only copies of a file

system, Oct. 6, 1998, U.S. Pat. No. 5,644,751, and Burnett, et. al., Jul. 1, 1997, Distributed file system (DFS) cache management system based on file access characteristics, discloses methods for implementing distributed file systems. U.S. Pat. No. 5,649,105, Aldred, et. al., Jul. 15, 1997, Collaborative working in a network, discloses programming methods for distributed applications using file sharing. U.S. Pat. No. 5,701,516, Chen, et. al., Dec. 23, 1997, High-performance non-volatile RAM protected write cache accelerator system employing DMA and data transferring scheme, discloses optimization methods for network attached hosts. However, those systems support only network file systems. Those systems do not support I/O channels.

In another application of storage systems, U.S. Pat. No. 5,790,795, Hough, Aug. 4, 1998, Media server system which employs a SCSI bus and which utilizes SCSI logical units to differentiate between transfer modes, discloses a media server that supports different file systems on different SCSI channels. However the system above is limited to a video data and does not support network attached hosts. Furthermore, in storage industry papers, Data Sharing, by Neema, Storage Management Solutions, Vol. 3, No. 3, May, 1998, and another industry paper, Storage management in UNIX environments: challenges and solutions, by Jerry Hoetger, Storage Management Solutions, Vol. 3, No. 4, survey a number of approaches in commercial storage systems and data sharing. However, existing storage systems are limited when applied to support multiple platform systems.

Therefore, a need exists to provide a high-performance data storage system that is assembled out of standard modules, using off-the-shelf hardware components and a standard general-purpose operating system that supports standard network software and protocols. In addition, the needs exists to provide a cached data storage system that permits independent data accesses from I/O channel attached local hosts, network attached remote hosts, and network-attached remote data storage systems.

SUMMARY OF THE INVENTION

The primary object of the invention is to provide a high performance, scalable, data storage system using off-the-shelf standard components. The preferred embodiment of the present invention comprises a network of PCs including an I/O channel adapter and network adapter and method for managing distributed cache memory stored in the plurality of PCs interconnected by the network. The use of standard PCs reduces the cost of the data storage system. The use of the network of PCs permits building large, high-performance, data storage systems.

Another object of the invention is to provide a distributed cache that supports arbitrary reads and writes arriving via I/O channels or network links, as well as a method for sharing data between two or more heterogeneous host computers using different data formats and connected to a data storage system. The method includes a translation module that inputs a record in a format compatible with the first host and stores the translated record in a data format compatible with the second host. Sharing of data in one format and having a translation module permitting representations in different formats in cache memory provides a means for improving performance of I/O requests and saving disk storage space.

In accordance with a preferred embodiment of the invention, a data storage system comprises a network of PCs each of which includes a cache memory, an I/O channel adapter for transmitting data over the channel and a network adapter for transmitting data and control signals over the network. In one embodiment, a method for managing resources in a cache memory ensures consistency of data stored in the distributed cache. In another embodiment, a method for sharing data between two or more heterogeneous hosts includes the steps of: reading a record in a format compatible with one computer; identifying a translation module associated with the second computer; translating the record into the format compatible with the second computer and writing said translated record into a cache memory.

The preferred embodiment of the present invention involves a method for building a data storage system that provides superior functionality at lower cost when compared to prior art systems. The superior functionality is achieved by using an underlying general-purpose operating system to provide utilities for managing storage devices, backing data, troubleshooting storage devices and performance monitoring. The lower cost is achieved by relying on standard components. Furthermore, the preferred embodiment of the present invention overcomes the limitations of prior art systems by providing concurrent access for both I/O channel attached hosts and network link attached hosts.

The preferred embodiment of this invention uses SCSI channels to connect to local hosts and uses standard network links card such as Ethernet, or ATM to connect to remote hosts. The alternate embodiment of the present invention uses fiber channel link such as Fibre Channel as defined by the Fibre Channel Association, FCA, 2570 West El Camino Real, Ste. 304, Mountain View, Calif. 94040-1313 or SSA as defined SSA Industry Association, DEPT 1165/B-013 5600 Cottle Road, San Jose, Calif. 95193. Prior art systems such as U.S. Pat. No. 5,841,997, Bleiwess, et. al., Nov. 24, 1998, Apparatus for effecting port switching of fibre channel loops, and U.S. Pat. No. 5,828,475, Bennett, et. al., Oct. 27, 1998, Bypass switching and messaging mechanism for providing intermix fiber optic switch using a bypass bus and buffer, disclosure methods that connects disks and controllers. However, the problems remain in software, solution of which require methods described in the preferred embodiment of the present invention.

The drawings constitute a part of this specification and include exemplary embodiments to the invention, which may be embodied in various forms.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows data storage systems configurations;

FIG. **2** illustrates in block diagram form the alternate embodiment of the data storage system of the present invention;

FIG. **2**A illustrates in block diagram form the alternate embodiment of the data storage system of the present invention;

FIG. **2**B illustrates in block diagram form another variation of the alternate embodiment of the present invention;

FIG. **3** shows a PC data storage system;

FIG. **4** illustrates in data flow diagram form the operations of a data storage system including: FIG. **4**A illustrating operations in write exclusive mode, FIG. **4**B in read exclusive mode, FIG. **4**C in write shared mode, FIG. **4**D in read shared mode, FIG. **4**E in disk interrupt, FIG. **4**F in page flusher; and

FIG. **5** illustrates in block diagram form data sharing operations.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Detailed descriptions of the preferred embodiment are provided herein. It is to be understood, however, that the present invention may be embodied in various forms. Therefore, specific details disclosed herein are not to be interpreted as limiting.

FIG. **1** illustrates data storage system configurations of the preferred embodiment. The PC data storage system **131** services a plurality of channel attached host processors **111**, **112** using channels **121**, **122**, and a plurality of network attached host processors **106**, **107** using network link **151**, and a plurality of network attached data storage systems **132**, **133** using network links **152**, **153**. PC storage system **132** services channel attached hosts **157**, **158**.

Hosts **157** and **158** access a data storage system **131** indirectly via network attached data storage system **132**, thereby offloading communications protocol overhead from remote hosts **157**, **158**. Hosts **106** and **107** directly access storage system **131** via network link **151** thereby incurring communications protocol overhead on hosts **106**, **107** and therefore decreasing performance of applications running on said hosts.

Host **111** accesses remote disk **181** via local data storage system **131**, network link **153**, and remote data storage system **133** without incurring protocol overhead on host **111**. Host **157** accesses disk **161** via data storage system **133**, network link **152**, and data storage system **131** without incurring protocol overhead on host **157**. Host **106** directly accesses local disk **161** via network link **151** thereby incurring protocol overhead. The disks **191**, **192** that are attached to hosts **106**, **107** without a data storage system, cannot be accessed by outside hosts.

The preferred embodiment of the present inventions uses well-established technologies such as SCSI channels for I/O traffic and Ethernet link for network traffic. In FIG. **2**, the alternate embodiment of the present invention uses fiber channel technology for both I/O traffic and network traffic. The fiber channel connects computers and hard disks into one logical network. In one variation of the alternate embodiment in FIG. **2**, the fiber optics link is organized as a Fiber Channel Arbitrated Loop (FCAL). In another variation shown in FIG. **2**A, the fiber optics link is organized as a switching network. In yet another variation in FIG. **2**B, the fiber channel is organized in two FCAL loops connected via switch.

FIG. **3** shows a software architecture and modules of a PC data storage system corresponding to the data storage system **131** in FIG. **1**. Data is received from the hosts **111**, **112** via I/O channels **121**, **122** in front-end software module **310** in FIG. **3**. The front-end module **310** handles channel commands and places the results in cache memory **322** in the form of new data or modification to data already stored on the disk **161**. The cache manager software module **320** calls routines in the configuration manager **340** to ensure consistency of the cache memory in other network attached data storage systems. At some later point in time, the back-end software module **342** invokes a page flusher module to write modified data to disks **161** and **162** and free up cache memory.

In FIG. **3**, front-end module **310** including I/O adapter driver has been modified to accept target SCSI I/O requests from hosts **111** and **112**. Said front-end module handles I/O requests in such a manner that hosts **111** and **112** are not aware of a data storage system. Hosts **111** and **112** issue I/O requests as if the request is going to a standard disk.

The presence of fast access cache memory permits front end channels and network links to operate completely independent of the back-end physical disk devices. Because of this front-end/back-end separation, the data storage system **131** is liberated from the I/O channel and network timing dependencies. The data storage system is free to dedicate its processing resources to increase performance through more intelligent scheduling and data transfer network protocol.

FIG. **4** shows a flowchart of a data storage system in the process of reading or writing to data volumes stored on disk drives shown in FIG. **3**. The flowchart uses a volume access table **450** (see also FIG. **5**) and controlled by the configuration manager. Local operations begin in step **401** where the corresponding front-end module **310** of FIG. **3** allocates a channel and waits for I/O requests from the initiating hosts **111** or **112**. Remote operations begin in step **402**. Depending upon the status of the value in a volume access table **450** the requests are routed either as shown in FIG. **4**A for write exclusive mode, FIG. **4**B for read exclusive, FIG. **4**C for write shared or FIG. **4**D for read shared. Concurrently with the processing of I/O operations, the independent page flusher daemon shown in FIG. **4**F scans cache memory and writes buffers to disks. Disk interrupt processing is shown in FIG. **4**E.

Volume access table **450** (see FIG. **4**) contains a mapping between hosts and volumes specifying an access mode value. If the access mode is set neither to shared nor exclusive, the configuration manager forwards I/O requests directly to disk. In addition to the access mode, the volume access table may contain other values that help the configuration manager and improve performance of the data storage system.

In another embodiment of this application, shown in FIG. **5**, the volume access table includes a translation module for a given host to facilitate volume mapping. The translation module is a dynamically loadable library that can be changed, compiled and linked at run-time.

A user of a data storage system can externally set the values and parameters in a volume access table. For each host and volume pair, a user can explicitly specify the access mode value. For some applications, where data on a remote volume is accessed infrequently, the user may want to specify other than shared or exclusive in order to disable cache for the remote volume. By disabling caching, the user eliminates cache coherency traffic entirely for the volume. In a data storage system, a user or a system administrator actively monitors and changes the behavior of a cache manager by changing values in a volume access table in order to improve performance of the data storage system.

FIG. **4**A shows a flowchart of the cache manager **320** (see FIG. **3**) as it processes a write request in an exclusive mode. In step **411** of FIG. **4**A, the cache manager checks whether the requested buffer is in cache or not. For a cache miss, in step **412**, the cache manager allocates a new buffer for storing data that will be written. For a cache hit, the cache manager branches directly to step **413** where data is copied into the newly allocated buffer. In step **414**, the cache manager calls a configuration manager routine that sends an invalidate request to the list of shared hosts for this particular volume. In step **415**, the cache manager checks the type of a request. For a channel type of a request, the cache manager returns to step **405** to release the channel. For a network type of a request, the cache manager proceeds to release network request in step **419** on the right side of FIG. **4**A.

On the right side of FIG. **4**A, in step **416**, network interrupt identifies and receives a remote write request. In step **417**, the cache manager calls configuration manager

routine to determine the validity of the request. Bad requests are ignored in step **418**. Correct requests proceed to step for **410** for write exclusive processing. Step **415** returns the flow to step **419**, which releases network resources.

FIG. **4**B shows a flowchart of the cache manager as it processes a read request in an exclusive mode. In step **420**, the cache manager checks whether the requested buffer is in cache or not. For a cache miss, in step **421**, the cache manager allocates a buffer for storing data that will be read in. In step **422**, the cache manager updates the buffer status with read pending. In step **423**, the cache manager starts an operation to read from a hard disk driver and proceeds to release the channel in step **405**. For a cache hit, in step **424**, the cache manager transmits read data and proceeds to release the channel in step **405**. For an identified network request, in step **425**, the cache manager sends back read results in step **429**.

On the right side of FIG. **4**B, in step **426**, network interrupt identifies and receives a remote read request. In step **427**, the cache manager calls a configuration manager routine that checks the configuration file and ignores bad requests in step **428**. Correct requests proceed to step **420** for read exclusive processing. Step **425** returns the flow to step **429** that sends read results.

FIG. **4**C shows a flowchart of the cache manager as it processes a write request in a shared mode. In step **430**, the cache manager checks whether the requested buffer is in cache or not. For a cache miss, in step **431**, the cache manager allocates a new buffer for storing data that will be written. For a cache hit, the cache manager branches directly to step **432** where data is copied into the newly allocated buffer. In step **433**, the cache manager updates the buffer status with write pending and proceeds to step **434** to release the channel. In step **435**, the cache manager calls a configuration manager routine that sends a remote write request to the host that holds this particular volume in an exclusive mode. In follow up to step **435**, the cache manager returns to the beginning of FIG. **4**.

On the right side of FIG. **4**C, the cache manager updates the buffer status with write done in step **444**. The flow begins with the network interrupt that calls configuration manager to validate the request in step **441**. Bad requests are ignored in step **442**. A correct request proceeds to step **443** that checks whether the status of this particular buffer is write pending. If the status is pending, in step **444**, the cache manager updates the buffer status to write done. For any other buffer status, in step **445**, the cache manager updates the status to free. This buffer is released in accordance with the invalidate request that has come from a remote host that holds this volume in an exclusive mode as has been described in FIG. **4**A.

FIG. **4**D shows a flowchart of the cache manager as it processes a read request in a shared mode. In step **450**, the cache manager checks whether the requested buffer is in cache or not. For a cache miss, in step **452**, the cache manager allocates a buffer for storing data that will be read into. For a cache hit, in step **451**, the cache manager transmits read data and proceeds to step **405** to release the channel. In the case of the cache miss, the cache manager allocates a new buffer in step **452** and updates its status to read pending in step **453**. In step **454**, the cache manager closes the channel with an optimizer that maintains a pool of open channels which are kept open only for the specified amount of time. In step **455**, the cache manager calls configuration manager routine that sends a remote read request to the host that holds this particular volume in an exclusive mode. The operations of the host holding volume in read exclusive mode have been shown in FIG. **4**B.

On the right side of FIG. **4**D, in step **456**, a network interrupt identifies a remote read result. In step **457**, the cache manager performs an optimized channel open. Depending upon the status of the optimizer that has been initiated in step **454**, the cache manager may immediately get access to the still open channel or, if the optimizer fails, the cache manager may need to reopen the channel. In step **458**, the cache manager transmits read data. In step **459**, the cache manager updates the buffer status to read done and proceeds to step **459** where it releases the channel.

FIG. **4**E shows a flowchart of the cache manager as it processes a hard disk interrupt request marking the completion of a read or write request. The read request has been started in step **423** in FIG. **4**B. The write request has been started in step **475** in FIG. **4**F. In step **460**, the cache manager checks the type of the hardware interrupt. For a write interrupt in step **461**, the cache manager updates the buffer status to write done and releases resources associated with the interrupt. For a read interrupt in step **462**, the cache manager updates the buffer status to read done. In step **463**, the cache manager checks request type of the read operation that has been started in FIG. **4**B. For a channel request, the cache manager proceeds to open a channel in step **466**. In step **467**, the cache manager transmits read data and proceeds to release the channel in step **405**. For a network request in step **464**, the cache manager finds the remote read requests that initiated the request. In step **466**, the cache manager sends read results and ends interrupt processing.

FIG. **4**F shows a flowchart of a cache memory page flusher. The flusher is a separate daemon running as part of the cache manager. In step **471**, the flusher waits for the specified amount of time. After the delay in step **472**, the flusher begins to scan pages in cached memory. In step **473**, the flusher checks the page status. If the page list has been exhausted in branch no more pages, the flusher returns to step **471** where it waits. If the page status is other than the write pending, the flusher returns to step **472** to continue scanning for more pages. If the page status is write pending, the flusher proceeds to step **474**. In step **474**, the flusher checks the request type. For a channel type, the flusher starts a read operation in step **475** and returns to scan pages in step **472**. For a network type, the flusher checks for the network operations in progress and returns to step **472** for more pages.

FIG. **5** shows a data sharing operation between a plurality of heterogeneous host computers. In one embodiment the plurality of hosts includes but is not limited to a Sun Solaris workstation **111**, Windows NT server **112**, HP UNIX **106**, and Digital UNIX **107** each accessing a distinct virtual device respectively **510**, **520**, **530** and **540**. Configuration manager **560** provides concurrency control for accessing virtual devices that are mapped to the same physical device **161**. The configuration manager uses a volume access table **450** that has been shown in FIG. **4**.

A virtual device is a method that comprises three operations: initialization, read and write. The initialization operation registers a virtual device in an operating system on a heterogeneous host. Following the registration, the virtual device appears as if it is another physical device that can be brought on-line, offline or mounted on a file system. An application program running on the host cannot distinguish between a virtual device and a physical device.

For a virtual device, the read operation begins with a read from a physical device followed by a call to a translation module. The translation, module inputs a shared record in a

original format used on a physical disk and outputs the record in a new format that is specified for and is compatible with a host computer. The write operation begins with a call to a translation module that inputs a record in a new format and outputs a record in a shared format. The translation module is a dynamically loadable library that can be changed, compiled and linked at run-time.

The virtual device method described above allows a plurality of heterogeneous host computers to share one copy of data stored on a physical disk. In a data storage system using said virtual device method, a plurality of virtual devices is maintained in cache without requiring a copy of data on a physical disk.

While the invention has been described in connection with a preferred embodiment, it is not intended to limit the scope of the invention to the particular form set forth.

What is claimed is:

1. An apparatus comprising:
- a first interface configured to receive input/output (I/O) traffic from a first host device via a dedicated I/O channel, the I/O traffic comprising a read command;
- a second interface configured to receive first data via a network;
- a cache memory configured to store second data;
- a storage device configured to store third data; and
- a processor coupled to the cache memory, the processor coupled to the storage device via a communication path that is distinct from the dedicated I/O channel, the processor configured to access the cache memory during processing of the I/O traffic, the processor further configured to perform an access operation at the storage device based on the I/O traffic.

2. The apparatus of claim **1**, wherein the dedicated I/O channel is associated with dedicated throughput that corresponds to the I/O traffic.

3. The apparatus of claim **1**, wherein the I/O traffic is distinct from the first data.

4. The apparatus of claim **1**, wherein the dedicated I/O channel comprises a small computer system interface (SCSI) channel.

5. The apparatus of claim **1**, wherein the second interface comprises an Ethernet interface or an asynchronous transfer mode (ATM) interface.

6. The apparatus of claim **1**, wherein the I/O traffic further comprises a write command.

7. The apparatus of claim **1**, wherein the first data comprises a second read request, a write request, or a combination thereof.

8. The apparatus of claim **1**, wherein the network comprises a plurality of interconnected computing devices.

9. The apparatus of claim **1**, wherein the storage device and the dedicated I/O channel are independently accessible, and wherein the processor is configured to read at least a portion of the second data from the cache memory based on the read command.

10. The apparatus of claim **1**, wherein the processor is further configured to route the read command to the cache memory or to the storage device.

11. The apparatus of claim **1**, further comprising:
- configuration manager circuitry configured to route an I/O request included in the I/O traffic to the cache memory, to route the I/O request to the storage device, or to deny the I/O request;
- front-end circuitry configured to process the I/O request; and
- back-end circuitry configured to perform a read operation or a write operation at the storage device based on the I/O request.

12. The apparatus of claim **1**, wherein the host device comprises a server.

13. A method comprising:
- receiving input/output (I/O) traffic from a host device via a dedicated I/O channel at a first interface, the I/O traffic comprising a write command;
- receiving first data via a network at a second interface;
- storing second data at a cache memory;
- storing third data at a storage device;
- accessing the cache memory during processing of the I/O traffic; and
- performing one or more access operations at the storage device based on the I/O traffic, the one or more access operations utilizing a communication path between a processor and the storage device, the communication path distinct from the dedicated I/O channel.

14. The method of claim **13**, wherein the I/O traffic is distinct from the first data.

15. The method of claim **13**, further comprising performing a first access operation at the storage device independently of the dedicated I/O channel.

16. The method of claim **13**, wherein the second data is written at the cache memory in response to the write command.

17. The method of claim **13**, further comprising:
- receiving a read request via the network; and
- transmitting at least a portion of the second data from the cache memory via the network responsive to the read request being associated with a cache hit.

18. The method of claim **13**, further comprising storing fourth data at the cache memory, wherein the second data is indicated by the write command, and wherein the fourth data is indicated by a second write command received at the second interface.

19. An apparatus comprising:
- a first interface configured to receive input/output (I/O) traffic from a host computer via a dedicated I/O channel, the I/O traffic comprising one or more read commands, one or more write commands, or a combination thereof;
- a second interface configured to receive data via a network;
- a cache memory;
- a storage device; and
- a processor coupled via a communication path to the storage device, the processor configured to access the cache memory during processing of the I/O traffic, the processor further configured to perform one or more access operations at the storage device based on the I/O traffic, wherein the communication path is distinct from the dedicated I/O channel.

20. The apparatus of claim **19**, wherein the dedicated I/O channel is associated with dedicated throughput that corresponds to the I/O traffic.

21. The apparatus of claim **19**, wherein the I/O traffic is distinct from the data.

22. The apparatus of claim **19**, wherein the data corresponds to one or more read requests, one or more write requests, or a combination thereof.

23. The apparatus of claim **19**, wherein the storage device and the dedicated I/O channel are independently accessible.

24. An apparatus comprising:

means for receiving input/output (I/O) traffic from a first host device via a dedicated I/O channel, the I/O traffic comprising a write command;

means for receiving data via a network;

means for short-term data storage;

means for long-term data storage; and

means for performing one or more access operations at the means for short-term data storage during processing of the I/O traffic and for performing one or more access operations at the means for long-term data storage based on the I/O traffic, the means for performing distinct from the dedicated I/O channel.

* * * * *